Intel® Itanium® Processor

Specification Update

May 2003

Notice: The Intel® Itanium® processor may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are documented in this specification update.

Order Number: 249720-09

Contents

Revision History

Date	Version	Description
May 2003	009	Added erratum 40.
January 2003	008	Added C2 stepping and PAL version 8.8.30. Added errata 37-39.
August 2002	007	Updated workaround for erratum 28. Added errata 34-36.
February 2002	006	Updated workaround for errata 14 and 24. Added C1 stepping and PAL version 7.7.27 and 7.7.28. Added errata 32-33.
December 2001	005	Updated workaround for errata 14 and 24. Updated fix status for erratum 24. Added PAL version 6.6.26. Added errata 29-31.
September 2001	004	Updated workaround for erratum 9. Added PAL version 6.6.25. Added erratum 28.
August 2001	003	Updated workaround for erratum 24. Added PAL version 6.6.24. Added errata 25-27.
July 2001	002	Updated status for erratum 4 and erratum 10. Updated workaround for erratum 14. Added errata 22-24.
June 2001	001	This document is the first specification update for the Intel® Itanium® processor.

Preface

This document is an update to the specifications contained in the Affected Documents/Related Documents table below. This document is a compilation of device and documentation errata, specification clarifications, and changes. It is intended for hardware system manufacturers and software developers of applications, operating systems (OSs), or tools.

This document may also contain information that was not previously published.

Affected Documents/Related Documents

Title	Document #
Intel® Itanium™ Processor at 800 MHz and 733 MHz Datasheet	249634
Intel® Itanium™ Processor Hardware Developer's Manual	248701
Intel® Itanium® Architecture Software Developer's Manual, Volume 1: Application Architecture	245317
Intel® Itanium® Architecture Software Developer's Manual, Volume 2: System Architecture	245318
Intel® Itanium® Architecture Software Developer's Manual, Volume 3: Instruction Set Reference	245319
Intel® Itanium™ Processor Reference Manual for Software Development	245320
Intel® Itanium® Architecture Software Developer's Manual Specification Update	248699
Itanium® Processor Family System Abstraction Layer Specification	245359

Nomenclature

S-Spec Number is used to identify products. Products are differentiated by their unique characteristics, e.g. core speed, L3 cache size, package types, etc. Care should be taken to read all notes associated with each S-Spec number.

Errata are design defects or errors. Errata may cause the Intel® Itanium® processor's behavior to deviate from published specifications. Hardware and software designed to be used with any given processor must assume that all errata documented for that stepping are present on all devices unless otherwise noted.

Specification Changes are modifications to the current published specifications for the Itanium processor. These changes will be incorporated in the next release of the specifications.

Specification Clarifications describe a specification in greater detail or further highlight a specification's impact to a complex design situation. These clarifications will be incorporated in the next release of the specification.

Documentation Changes include typos, errors, or omissions from the current published specifications. These changes will be incorporated in the next release of the specifications.

Note: Errata remain in the specification update throughout the product's lifecycle, or until a particular stepping is no longer commercially available. Under these circumstances, errata removed from the specification update are archived and available upon request. Specification changes, specification clarifications, and documentation changes are removed from the specification update when the appropriate changes are made to the appropriate product specification or user documentation (datasheets, manuals, etc.).

Summary Table of Changes

The following table indicates the errata, specification changes, specification clarifications, or documentation changes which apply to the Itanium processor. Intel may fix some of the errata in a future stepping of the component or in a future release of the Processor Abstraction Layer (PAL), and account for the other outstanding issues through documentation or specification changes as noted. This table uses the notations indicated below.

Codes Used in Summary Table

Stepping

X:	Errata exists in the stepping or PAL version indicated. Specification Change or Clarification that applies to this stepping.
(No mark) or (Blank box):	This erratum is fixed in listed stepping or specification change does not apply to listed stepping or PAL version.

Page

(Page):	Page location of item in this document.

Status

Doc:	Document change or update will be implemented.
Plan fix:	This erratum may be fixed in a future stepping of the component, or in a future release of PAL.
Fixed:	This erratum has been previously fixed.
NoFix:	There are no plans to fix this erratum.

Row

	Change bar to left of table row indicates this erratum is either new or modified from the previous version of this document.

Errata (Sheet 1 of 2)

No.	Processor Stepping			PAL Versions								Status	Errata
	C0	C1	C2	6.6.21	6.6.23	6.6.24	6.6.25	6.6.26	7.7.27	7.7.28	8.8.30		
1	X	X	X									NoFix	IFA may contain incorrect address
2	X	X	X									NoFix	AR.ITC returns an incorrect value
3	X	X	X									NoFix	L1D line fill during DBR/IBR access may result in an incorrect line fill
4	X	X	X									NoFix	IA-32: Incorrect self-modifying code behavior
5	X	X	X									NoFix	System bus pins driven low during JTAG continuity testing
6	X											Fixed	INIT# signal not recognized properly
7				X								Fixed	BINIT condition prevents error logging
8	X	X	X									NoFix	Internal BINIT during low power light halt mode
9	X	X	X									NoFix	Incorrect error logging information for double-bit ECC error on PTC or interrupt transaction
10	X	X	X									NoFix	Processor hang during nested BINIT
11				X	X							Fixed	False BIL transactions during PAL_CACHE_FLUSH
12	X											Fixed	ALL_STOPS_DISPERSED and EXPL_STOPS_DISPERSED not operating correctly
13	X	X	X									NoFix	Snoop hit to modified line in L2 cache with tag parity error
14	X											Fixed	Infinite DBSY# hang due to livelock condition
15	X	X	X									NoFix	Unexpected writeback transaction in 2:11 mode with bus parking enabled
16	X											Fixed	IA-32: Code with FP instruction followed by integer instruction with interrupt pending may not execute correctly
17				X								Fixed	Bus parking always enabled and not controllable by PAL
18				X	X							Fixed	IA-32: FSINCOS may not generate FP precision exception
19	X	X	X									NoFix	chk.a.clr/ld.c.clr incorrectly clears its ALAT entry
20	X											Fixed	IA-32: Back to back semaphores under certain circumstances may cause processor to hang
21	X											Fixed	Modification of PFS under certain circumstances can lead to unexpected program behavior
22				X	X							Fixed	Corrected machine check not indicated when CMCI is promoted to MCA
23				X	X							Fixed	PAL_MC_ERROR_INFO may return incorrect target address information
24	X	X	X									NoFix	Incorrect store update to L1D cache under certain circumstances
25	X	X	X									NoFix	IA-32: unexpected page fault exception on SETcc instruction
26	X	X	X									NoFix	IA-32: PUNPCKLBW/PUNPCKLWD/PUNPCKLDQ reads 8 bytes instead of 4 bytes
27	X	X	X									NoFix	Processor may incorrectly report an IA-32 BIST failure
28	X											Fixed	Back-to-back pair of loads to L1D cache result in possible livelock condition
29				X	X	X	X	X	X	X		Fix	PAL_PERF_MON_INFO returns incorrect information on registers which can count retired instruction bundles
30	X	X	X									NoFix	ALAT not guaranteed to be invalidated by an external snoop under specific conditions
31	X	X	X									NoFix	WC store may be dropped under certain conditions
32								X				Fixed	PAL 6.6.26 incorrectly reports a BIST failure on C-1 stepping processors

Errata (Sheet 2 of 2)

No.	Processor Stepping			PAL Versions								Status	Errata
	C0	C1	C2	6.6.21	6.6.23	6.6.24	6.6.25	6.6.26	7.7.27	7.7.28	8.8.30		
33	X	X	X									NoFix	BSP may be incorrectly updated after br.ret under certain conditions
34	X	X										Fixed	Specific code sequence involving FP check load may not execute correctly under certain conditions
35	X	X	X									NoFix	RFI to non-slot 0 B-syllable may result in incorrect software debugger operation
36	X	X	X									NoFix	Removal of WAW hazard may lead to undefined result
37	X	X	X									NoFix	Specific code sequence involving br.ret followed by mov from AR.EC may return incorrect EC value
38	X	X	X									NoFix	Specific code sequence involving bsw followed immediately by loadrs may not execute correctly
39				X	X	X	X	X	X	X	X	NoFix	PAL_TEST_PROC may cause unexpected system behavior
40												Fixed	FPSWA version 1.12 may overwrite register fr12[a]

a. This erratum applies to FPSWA version 1.12 only.

Specification Changes

No.	Processor Stepping			PAL Versions								Status	SPECIFICATION CHANGES
	C0	C1	C2	6.6.21	6.6.23	6.6.24	6.6.25	6.6.26	7.7.27	7.7.28	8.8.30		
													None for this revision of this Specification Update

Specification Clarifications

No.	Processor Stepping			PAL Versions								Status	SPECIFICATION CLARIFICATIONS
	C0	C1	C2	6.6.21	6.6.23	6.6.24	6.6.25	6.6.26	7.7.27	7.7.28	8.8.30		
													None for this revision of this Specification Update

Documentation Changes

No.	Processor Stepping			PAL Versions								Status	DOCUMENTATION CHANGES
	C0	C1	C2	6.6.21	6.6.23	6.6.24	6.6.25	6.6.26	7.7.27	7.7.28	8.8.30		
													None for this revision of this Specification Update

Identification Information

Intel® Itanium® Processor Cartridge Marking

The cartridge mark for the product is a laser marked label attached to the end of the PAC418 cartridge opposite the power tab. The mark provides the following information:

- Product Brand Name
- Manufacture Traceability (including manufacturing lot and unit identification)
- Manufacturing Site
- Mask Work and Copyright Protection for the Processor and Cache Die
- Product Identification (including processor core speed, cache size, bus speed)
- Intel Company Logo
- Two-dimensional Product Identification Matrix (Intel internal use)

FFFFFFFF - manufacture traceability number
NNNN - serial number
yy - year

Intel® Itanium® Processor Identification and Package Information

S-Spec/QDF Number	Core Stepping	CPUID[a]	Speed (MHz)	L3 Size (Mbytes)
SL4LT	C0	0007000604h	733/133	2
SL4LS	C0	0007000604h	733/133	4
SL4LR	C0	0007000604h	800/133	2
SL4LQ	C0	0007000604h	800/133	4
SL5VS	C1	0007000704h	733/133	2
SL5VT	C1	0007000704h	733/133	4
SL5VU	C1	0007000704h	800/133	2
SL5VW	C1	0007000704h	800/133	4
SL6RH	C2	0007000804h	733/133	2
SL6RK	C2	0007000804h	800/133	2
SL6RL	C2	0007000804h	800/133	4

a. The CPUID column in this table indicates the contents of bits 39:0 of CPUID Register 3. Bits 63:40 of this register are reserved.

Mixed Steppings in MP Systems

The Itanium processor supports multi-processor platforms using mixed processor steppings of N and N-1 on the same system bus. While Intel has done nothing to specifically prevent processors within a multi-processor environment from operating with mixed steppings beyond the N and N-1 configurations, there may be uncharacterized errata that exist in such configurations.

Note: The Itanium processor C-2 stepping will not be tested in a mixed processor stepping configuration to support a combination of N (C-2 stepping) and N-1 (C-1 stepping) on the same system bus within a multiprocessor environment.

Errata

1. IFA may contain incorrect address

Problem: The Interruption Fault Address (IFA) is not correctly reported for Instruction Access faults and Instruction Debug faults when the processor is executing Itanium instructions. The IFA is reported correctly when the processor is executing IA-32 instructions.

Implication: The IFA may report a wrong IP address on an Instruction Access-bit access or an Instruction Debug fault. OSs should not rely on the value in IFA when these faults occur.

Workaround: The Interruption Instruction Bundle Pointer (IIP) provides the same information as the IFA for Instruction Access faults and Instruction Debug faults. These fault handlers should use the IFA or the IIP based on IPSR.is:

```
if (IPSR.is)
       use IFA
else
       use IIP
```

Status: For the steppings affected, see the Summary Table of Changes.

2. AR.ITC returns an incorrect value

Implication: The 64-bit Interval Timer Counter (AR.ITC) register may return an incorrect value when the lower 32-bits are all F's. In this case, the value returned in the upper 32-bits is incremented by 1. For example, when the value returned is 0x1FFFFFFFF, the actual value should be 0x0FFFFFFFF.

Implication: Software that utilizes the AR.ITC register will receive an incorrect value in this case.

Workaround: The workaround is for software to re-read the AR.ITC register when it detects all F's in the lower 32-bits.

Status: For the steppings affected, see the Summary Table of Changes.

3. L1D line fill during DBR/IBR access may result in an incorrect line fill

Implication: Under certain circumstances, if a line fill to the L1 data cache occurs simultaneously with Debug Breakpoint Register (DBR/IBR) accesses, the line fill to the L1 data cache may incorrectly occur to address zero. In general, this erratum can be encountered if the code sequence contains a 'Move to Data Breakpoint Register' instruction followed by a 'Move to Instruction Breakpoint Register' instruction.

Implication: Code that operates on the debug breakpoint registers in the above sequence may possibly overwrite address zero data. Subsequent use of this data may result in unpredictable behavior.

Workaround: Insert a serializing instruction '`srlz.d`' in between the '`mov dbr`' and the '`mov ibr`' instruction.

Status: For the steppings affected, see the Summary Table of Changes.

4. IA-32: Incorrect self-modifying code behavior

Problem: Under certain circumstances, the processor may fail to correctly execute IA-32 self-modifying code (SMC). In the failing scenario, the processor does not wait for a previous store to complete prior to issuing and completing an instruction fetch to the same address.

Implication: IA-32 SMC may execute incorrectly resulting in unexpected program behavior.

Workaround: A workaround for this erratum is implemented in PAL 6.6.21 and later versions.

Status: For the steppings affected, see the Summary Table of Changes.

5. System bus pins driven low during JTAG continuity testing

Problem: During JTAG Boundary Scan continuity testing, the system bus pins may be driven low by the processor not allowing the pins to be forced to a high state.

Implication: The system bus RESET# pin cannot be tested for continuity as board-level component interconnect testing requires the system bus RESET# pin to be kept asserted.

Workaround: Assert the system bus RESET# pin during JTAG Boundary Scan continuity testing.

Status: For the steppings affected, see the Summary Table of Changes.

6. INIT# signal not recognized properly

Problem: The INIT# signal triggers an unmasked interrupt to the processor. When operating at the 2:11 bus-to-core frequency ratio, the assertion of the INIT# pin may not always be recognized by the processor, preventing the processor from taking the interrupt.

Implication: The processor may miss taking the INIT# interrupt when operating at the 2:11 bus-to-core frequency ratio.

Note: This erratum does not impact the use of the INIT# pin for power-on configuration during reset.

Workaround: Either a system bus-based interrupt transaction or the Platform Management Interrupt (PMI)# input can be used to implement the same functionality.

Status: For the steppings affected, see the Summary Table of Changes.

7. BINIT condition prevents error logging

Problem: On a BINIT non-recoverable Machine Check Abort (MCA) condition, the processor may enter an infinite loop in the PAL MCA handler. This prevents hand-off to the SAL MCA handler from occurring.

Implication: On a BINIT MCA condition, the processor may encounter a hang. This will prevent the BINIT error from being logged.

Note: Since BINIT indicates that a fatal condition has occurred which prevents reliable future operation, the system would normally be reset.

Workaround: None identified at this time.

Status: For the steppings affected, see the Summary Table of Changes.

8. Internal BINIT during low power light halt mode

Problem: The processor enters a light halt mode upon executing either the PAL_HALT_LIGHT or the PAL_HALT_LIGHT_SPECIAL calls. During light halt mode, if the processor encounters an internal BINIT (Bus Initialization) non-recoverable MCA condition, the processor may not flag the MCA as expected.

Implication: The processor may not flag the fatal error condition as expected during light halt mode.

Workaround: Flush and invalidate all cache lines by calling the PAL_CACHE_FLUSH call with inv = 1 before entering the light halt mode.

Status: For the steppings affected, see the Summary Table of Changes.

9. Incorrect error logging information for double-bit ECC error on PTC or interrupt transaction

Problem: The processor logs an incorrect request type and address information for a double-bit ECC error on an interrupt or an inbound ptc transaction.

Implication: For a double-bit ECC error on an inbound ptc transaction, the PAL machine check handler may incorrectly raise machine check abort. For a double-bit ECC error on an interrupt transaction, the processor may incorrectly raise a Corrected Machine Check Interrupt (CMCI) instead of an MCA, leading to possible incorrect functionality.

Workaround: The workaround implemented in PAL release 6.6.23 and 6.6.25 promotes all PAL continuable double-bit ECC system bus errors to OS-recoverable machine check aborts. The MCA logging for double-bit ECC system bus errors may be inaccurate, but errors will be contained.

Note: The workaround for this erratum is not included in PAL 6.6.24.

Status: For the steppings affected, see the Summary Table of Changes.

10. Processor hang during nested BINIT

Problem: On an external assertion of a BINIT non-recoverable MCA within a certain time interval while the processor is servicing a previous external or internally generated BINIT non-recoverable MCA, the system bus may stall due to an infinite assertion of the Block Next Request (BNR)# signal.

Implication: The processor may hang as a result of the nested BINIT condition.

Workaround: None identified at this time.

Status: For the steppings affected, see the Summary Table of Changes.

11. False BIL transactions during PAL_CACHE_FLUSH

Problem: During PAL_FLUSH_CACHE, the processor may issue BIL transactions to incorrect addresses when run in cacheable mode with inv = 1. Although the false BIL transaction is issued, actual data is not modified and cache coherency is still maintained.

Implication: False BIL transactions may be seen on the system bus during execution of the PAL_CACHE_FLUSH call.

Workaround: Since the false BIL always targets a certain address range, depending on how PAL is mapped to writeback memory, the chipset can ignore any bus transactions issued by the processor to that address range.

Status: For the steppings affected, see the Summary Table of Changes.

12. ALL_STOPS_DISPERSED and EXPL_STOPS_DISPERSED not operating correctly

Problem: The performance monitoring event ALL_STOPS_DISPERSED counts the implicit and explicit stops dispersed, while the event EXPL_STOPS_DISPERSED counts the explicit stops dispersed. These counters incorrectly count their respective events.

Implication: These performance monitoring events may report an incorrect count.

Workaround: None identified at this time.

Status: For the steppings affected, see the Summary Table of Changes.

13. Snoop hit to modified line in L2 cache with tag parity error

Problem: On a snoop hit to a modified line in the L2 cache which encounters a tag parity error, the processor may incorrectly report the snoop response as a miss and hang. The processor does flag an MCA and raise Bus Error (BERR) as expected.

Implication: The processor may hang as a result of encountering a snoop hit to a modified line in the L2 cache with a tag parity error. Also, since the L2 cache reports the snoop response as a miss, another processor can potentially modify the data for that cache line.

Workaround: Enable BERR to BINIT promotion using the PAL_PROC_SET_FEATURES call.

Status: For the steppings affected, see the Summary Table of Changes.

14. Infinite DBSY# hang due to livelock condition

Problem: In the event of several internal conditions and the specific alignment of these conditions, the processor may encounter a potential livelock. The conditions involve a stream of L3 cache traffic, victimization of modified lines from the L3 cache, a pending L2 cache fill, and a snoop request from an external bus agent that hits the same pending L2 cache line.

Implication: As a result of the livelock condition, the processor asserts the Data Bus Busy (DBSY#) signal without a corresponding data transfer, causing the system to hang. This erratum has only been observed running a synthetic cache stress test.

Workaround: The workaround for this erratum may be enabled with PAL 7.7.28 and later versions.

Status: For the steppings affected, see the Summary Table of Changes.

15. Unexpected writeback transaction in 2:11 mode with bus parking enabled

Problem: Following are the necessary conditions for this erratum:

1. The processor is configured in 2:11 bus-to-core ratio and bus parking is enabled.
2. One processor Px owns two cache lines A and B in the M-state.
3. A snoop request (due to BIL/BRIL transaction) for cache line A followed by a snoop request (due to BIL/BRIL transaction) for cache line B is made (by priority agent or processor Py).
4. A specific timing relationship of IDS#, ADS#, and BPRI#/BRy# signals needs to be met.
5. In parallel, Px tries to issue an explicit writeback for A followed by an explicit writeback for B.

As a result, the processor may incorrectly issue an explicit writeback transaction for B on the system bus, following the implicit writeback for A and B, which is a violation of the system bus protocol. In addition, the explicit writeback transaction for B may contain incorrect data.

If the processor has another read/write request that needs to be issued to the system bus immediately after the explicit writeback transaction for B, the request may be held pending indefinitely in the processor bus queue causing the processor to hang.

Implication: The processor may issue an explicit writeback transaction with incorrect data when it is not expected to do so. In the case where the read/write request is held pending indefinitely inside the processor, the system may hang. This erratum has only been observed running a synthetic stress test.

Workaround: The workaround for this erratum is to disable bus parking.

Status: For the steppings affected, see the Summary Table of Changes.

16. IA-32: Code with FP instruction followed by integer instruction with interrupt pending may not execute correctly

Problem: In the event of certain internal conditions, IA-32 code that contains a floating-point (FP) instruction followed immediately by an integer instruction with an interrupt pending may not execute correctly.

Implication: IA-32 FP code in the above scenario may not execute correctly.

Workaround: A workaround for this erratum is implemented in PAL release 6.6.23.

Status: For the steppings affected, see the Summary Table of Changes.

17. Bus parking always enabled and not controllable by PAL

Problem: The processor can be configured to not park on the request bus when idle, if A15# is sampled deasserted at the asserted-to-deasserted transition of RESET#. Due to this erratum, PAL overrides the hardware setting and enables bus parking by default. Additionally, the PAL_BUS_GET_FEATURES call reports that bus parking (bit 29) is not controllable and does not allow the PAL_BUS_SET_FEATURES call to enable or disable bus parking.

Implication: PAL enables bus parking by default and overrides the hardware setting at reset. Additionally, bus parking cannot be controlled using the PAL_BUS_SET_FEATURES call.

Workaround: None identified at this time.

Status: For the steppings affected, see the Summary Table of Changes.

18. IA-32: FSINCOS may not generate FP precision exception

Problem: The IA-32 FP instruction, FSINCOS computes both the sine and cosine of a source operand, with the final FP operations in the computation expected to generate a precision exception. For certain source operand values, these final operations may be computed exactly, causing the expected FP precision exception not to be generated.

Implication: For certain source operands, the FSINCOS instruction fails to generate the expected FP precision exception.

Workaround: None identified at this time.

Status: For the steppings affected, see the Summary Table of Changes.

19. chk.a.clr/ld.c.clr incorrectly clears its ALAT entry

Problem: Under certain conditions following the execution of an advanced load, the speculation check instructions, `chk.a.clr` and `ld.c.clr`, may incorrectly report a miss in the Advanced Load Address Table (ALAT) indicating that the data speculation was unsuccessful.

Implication: As a result of the miss, the check load (`ld.c`) will reload the correct value from memory and the advanced load check (`chk.a`) will branch to compiler-generated recovery code.

Workaround: Replace `chk.a.clr` with `chk.a.nc`, and `ld.c.clr` with `ld.c.nc`.

Status: For the steppings affected, see the Summary Table of Changes.

20. IA-32: Back to back semaphores under certain circumstances may cause processor to hang

Problem: In the event of certain internal conditions involving the processor Virtual Hash Page Table (VHPT) walker, while the processor is executing a sequence of back-to-back IA-32 semaphore operations, the processor may hang.

Implication: IA-32 code operating on locked semaphores in the above scenario may cause the processor to hang.

Note: Typical spin lock code containing semaphores is not affected by this erratum. This erratum has only been observed running a focused test in a system validation environment.

Workaround: Separate back-to-back IA-32 semaphores with a minimum of two NOPs.

Status: For the steppings affected, see the Summary Table of Changes.

21. Modification of PFS under certain circumstances can lead to unexpected program behavior

Problem: The Previous Function State (PFS) register is provided to accelerate procedure calling. Under certain circumstances involving a specific code sequence, and a set of internal architectural and timing conditions, modification of the PFS fields prior to `br.ret` may result in incorrect restoration of state on returning back to the caller procedure.

Implication: Execution of code that meets the above criteria may result in unexpected program behavior. The erratum is not exposed for typical code involving PFS restoration identical to the initial PFS value stored on a procedure call, or code that creates a new stack frame by loading a new CFM to PFS prior to `br.ret`. This erratum has only been observed running a focused test in a system validation environment.

Workaround: None identified at this time.

Status: For the steppings affected, see the Summary Table of Changes.

22. Corrected machine check not indicated when CMCI is promoted to MCA

Problem: When CMCIs are promoted to MCAs through the PAL_PROC_SET_FEATURES call, on a corrected machine check error it is possible for PAL to not set the “cm” bit of the Processor State Parameter (PSP) at PALE_CHECK exit to indicate that the machine check has been corrected.

Implication: The indication that a machine check error has been corrected may not be provided in this case.

Workaround: None identified at this time.

Status: For the steppings affected, see the Summary Table of Changes.

23. PAL_MC_ERROR_INFO may return incorrect target address information

Problem: The PAL_MC_ERROR_INFO call returns the error information on a processor machine check. When a system bus or an L3 cache machine checks occur, the PAL_MC_ERROR_INFO call may not report the target address correctly.

Implication: The target address information for system bus or L3 cache machine checks may be incorrectly logged.

Workaround: None identified at this time.

Status: For the steppings affected, see the Summary Table of Changes.

24. Incorrect store update to L1D cache under certain circumstances

Problem: In the case of back-to-back stores that hit the L1 data cache (L1D), it is possible under certain circumstances for the data corresponding to one of the stores to be overwritten by a later store before it is updated in the cache. The necessary conditions for this erratum involve a stream of stores that hit the L1D cache with address dependencies amongst them and require specific internal timing conditions to be met.

Implication: A store may be incorrectly overwritten before it is updated in the cache. This erratum has only been observed running a synthetic test in a system validation environment.

Workaround: The workaround can be enabled by SAL using PAL 6.6.21 and PAL 6.6.23 and is enabled by default in PAL 6.6.24 and later versions. For C-1 stepping, this workaround can be enabled by SAL using PAL 6.6.26 and is enabled by default in PAL 7.7.27 and later versions.

Status: For the steppings affected, see the Summary Table of Changes.

25. IA-32: Unexpected page fault exception on SETcc instruction

Problem: Under specific conditions, if an IA-32 `SETcc` instruction is operating on self-modifying code (SMC) and the memory being stored by the `SETcc` instruction encounters a segment limit violation, the limit violation exception may not be taken and instead a page fault exception is taken.

Implication: IA-32 applications that execute SMC and encounter a segment limit violation would result in a GP fault, which is a non-recoverable condition for the application. The page fault exception will result in the same behavior. This erratum has only been observed during random instruction testing in a system validation environment.

Workaround: None identified at this time.

Status: For the steppings affected, see the Summary Table of Changes.

26. IA-32: PUNPCKLBW/PUNPCKLWD/PUNPCKLDQ reads 8 bytes instead of 4 bytes

Problem: If the source data for the IA-32 MMX™ Technology instructions, PUNPCKLBW/PUNPCKLWD/PUNPCKLDQ comes from memory, the processor should access only 32-bits from memory. Due to this erratum, the processor always accesses 64-bit data from memory, even though only 32-bits are used (no operation is performed on the 4 extra bytes).

Implication: If the memory address being accessed is close to the segment limit, a limit violation exception may be generated. In the case where the 4-byte data being accessed is located at the end of a page and the next page is invalid, an unexpected page fault may be generated. This erratum has only been observed during random instruction testing in a system validation environment.

Workaround: None identified at this time.

Status: For the steppings affected, see the Summary Table of Changes.

27. Processor may incorrectly report an IA-32 BIST failure

Problem: If Built-in Self Test (BIST) is enabled through INIT# pin signalling at reset, the processor may incorrectly report an IA-32 BIST failure.

Implication: Even though BIST executes correctly, the processor may incorrectly report an IA-32 BIST failure.

Workaround: If BIST is enabled, the firmware (SAL), upon reading the BIST result, can ignore any IA-32 BIST failure.

Status: For the steppings affected, see the Summary Table of Changes.

28. Back-to-back pair of loads to L1D cache result in possible livelock condition

Problem: In the event of certain internal conditions, the processor may encounter a livelock during lookup in the L1 data cache. The livelock condition may occur if the processor is executing a code sequence which contains a back-to-back pair of load instructions with all the load addresses hitting the L1 data cache and there is a specific relationship between the virtual addresses for these loads.

Implication: Because the processor continues to take interrupts during the livelock, the livelock condition will likely be broken when an interrupt is received or when there is a context switch to another process. In the case where the livelock condition is not broken, the processor may hang and eventually time-out (if internal processor time-out is enabled) generating a BINIT. This erratum has only been observed running a synthetic test in a system validation environment.

Workaround: The workaround for this erratum may be enabled with PAL 6.6.26 and later versions.

Status: For the steppings affected, see the Summary Table of Changes.

29. PAL_PERF_MON_INFO returns incorrect information on registers which can count retired instruction bundles

Problem: The PAL_PERF_MON_INFO procedure is called to determine the number of performance monitors and the events which can be counted on the performance monitors. Due to this erratum, the PAL_PERF_MON_INFO procedure returns PMC4 as the only register available to count retired instruction bundles instead of returning PMC4 and PMC5.

Implication: The information returned by the PAL_PERF_MON_INFO call on the number of registers available for counting retired instruction bundles is incorrect.

Workaround: None identified at this time.

Status: For the steppings affected, see the Summary Table of Changes.

30. ALAT not guaranteed to be invalidated by an external snoop under specific conditions

Problem: Under certain conditions involving the timing of an advanced load instruction (`ld.a`) overlapping with an external snoop caused by a store (from another processor) to the same address as that of the advanced load, the ALAT may not get invalidated correctly by the external snoop.

Implication: As a result of this erratum, when operating in an MP environment, any advanced load boosted ahead of any acquire (including fences, mfa etc.) does not guarantee coherent ordering as shown in the code sequence below. This includes any advanced loads boosted ahead of any function call where the function may include memory ordering ops with acquire semantics:

```
Processor 0                                  Processor 1
ld.a X                                       st X
<any ordered op with acquire semantics>
ld.c X
```

Note: In the above example, the external snoop to processor 0 is a result of store to address X by processor 1.

The affected code sequence is not known by Intel to be generated in any current OS or compiler.

Workaround: To avoid this erratum, compilers and hand-written code should not boost an advanced load ahead of any ordered op with acquire semantics or ahead of opaque function calls.

Status: For the steppings affected, see the Summary Table of Changes.

31. WC store may be dropped under certain conditions

Problem: Under certain conditions involving Write Coalescing (WC) stores, a check load (`ld.c`) or an `lfetch` instruction interspersed with stores to WC space, may result in one of the WC stores to be dropped. This may occur in the following scenario:

- A store to WC space (A) is followed one or more instructions later by a `ld.c` or an `lfetch` to WC space which is to an address in the same WC buffer entries as store A.
- This `ld.c` (or `lfetch`) to WC space is followed by another WC store (B) either in the same instruction group, or in one instruction group later. Store B is to an address also in the same WC buffer entries as store A.

Implication: Code operating in WC space in the above scenario may not execute correctly.

Note: Typical usage of WC memory is by device driver code for graphics adaptors or graphics texture mapping code and consists mostly of store operations. This erratum has only been observed running a focused test in a system validation environment.

Workaround: None identified at this time.

Status: For the steppings affected, see the Summary Table of Changes.

32. PAL 6.6.26 incorrectly reports a BIST failure on C-1 stepping processors

Problem: When BIST is enabled on C-1 stepping processors, PAL 6.6.26 incorrectly reports a failing BIST result.

Implication: As a result of PAL incorrectly reporting a failing BIST result, the BIST result cannot be relied upon.

Workaround: Do not enable BIST on C-1 stepping processors when used with PAL 6.6.26.

Status: For the steppings affected, see the Summary Table of Changes.

33. BSP may be incorrectly updated after br.ret under certain conditions

Problem: Under certain conditions involving the operation of the Register Stack Engine (RSE) and branch prediction, the RSE Backing Store Pointer (BSP) may be incorrectly updated. The affected code sequence in this case involves the execution of a correctly predicted `br.ret` instruction which causes an underflow, and a RSE control instruction as the target of the `br.ret` instruction.

Implication: The BSP application register is incorrectly updated resulting in undefined behavior. The affected sequence is not known by Intel to be generated in any current OS or compiled code.

Workaround: None identified at this time.

Status: For the steppings affected, see the Summary Table of Changes.

34. Specific code sequence involving FP check load may not execute correctly under certain conditions

Under certain conditions involving the code sequence below, a FP check load to a FP register Fx, followed by another FP load to the same register Fx in the next instruction execution clock cycle may cause the consumer of Fx to get an obsolete value.

Note: In addition to the code sequence, certain timing-sensitive internal processor events also need to line up to cause the erratum to occur.

```
ldf.c Fx              //This check load needs to miss in the ALAT
                      //
<Fx consumer op>;;    //This instruction may use an obsolete value loaded by
                      //a previous advanced load
ldf Fx;;              //This can be either a regular/speculative floating
                      //point load or a setf instruction
```

Implication: Code that uses the above sequence may not execute correctly. The affected sequence is not known to be a common compiler-generated code but is considered to be theoretically possible where compilers more heavily utilize advanced loads and FP check loads. The sequence also involves a FP load operation to the same register as the one used by the previous check load which is typically not the case with compiled code since other registers are generally available for use.

Workaround: Compilers and assembly code can use one of the following workaround options:

1. Add a stop bit between the FP check load and the Fx consumer operation.
2. Add two stop bits (a NOP bundle) between the FP check load bundle and the subsequent bundle containing the FP load to the same register Fx.
3. Use different FP registers for the FP check load and the second FP load.
4. Only use the `chk.a` instruction on FP load speculation.

Status: For the steppings affected, see the Summary Table of Changes.

35. RFI to non-slot 0 B-syllable may result in incorrect software debugger operation

Problem: Under certain conditions, restarting instruction execution (after an RFI) at a branch syllable on slot1/slot2 may cause the branch syllable to vector to the wrong target address.

The following conditions are necessary for the erratum to manifest.

1. The RFI target bundle template is either MBB or BBB, and has two back to back IP-relative branch/calls (`br.cond/br.call`) syllables.

   Note: `br.ret`, `brl`, and *indirect_form* of `br.cond/br.call` are not affected.

2. The instruction restart after the RFI (indicated by `IPSR.ri`) occurs on the second B-syllable of the BB sequence. This means that `IPSR.ri` = 1 or 2 for a BBB bundle, or `IPSR.ri` = 2 for an MBB bundle.
3. The RFI target bundle encounters a Instruction Translation Lookaside Buffer (ITLB) miss, and the miss is satisfied by the VHPT walker.

Under these circumstances, the second B-syllable of the BB sequence may vector processor execution to the wrong target address.

Implication: This erratum may occur under a debugger when setting software breakpoints in MBB/BBB bundles, either with single step or instruction breakpoints.

Workaround: None identified at this time.

Status: For the steppings affected, see the Summary Table of Changes.

36. Removal of WAW hazard may lead to undefined result

Problem: Due to internal conditions an allowed WAW dependency may become a WAW hazard under the following circumstances:

- A move to the `AR.PFS` register is followed by a `BR.CALL` and both are executed in the same issue group, or
- A move to the `AR.EC` register is followed by a `BR.RET` and both are executed in the same issue group.

These combinations of instructions are legal WAW memory dependencies if one of the operations is predicated off. If preceding instructions (as indicated above) combine to change the predication on the `BR.CALL` or `BR.RET` from predicated true to predicated false, the processor may mistakenly decide the WAW hazard is still present and fail to recognize that the WAW has been removed which may result in an undefined value for `ar.pfs` or `ar.ec`.

The following code sequence demonstrates this issue:

```
p15 = 1;
;;
mov ar.pfs = R[x];
ld.c R[y] = [m]; //causes R[y] to be reloaded.

cmp.eq p15, p16 = R[y], R0;
(p15) br.call;
```

The RAW dependencies on `ld.c` to `cmp` and `cmp` to branch are legal. When the processor executes the issue group, the WAW hazard is present and the PFS results are undefined. If the `ld.c` misses the ALAT, the `cmp` to branch will be re-executed, the new result of the `ld.c` causes the p15 value to change to false and thus eliminate the WAW. Then the processor may fail to recognize that the WAW has been removed.

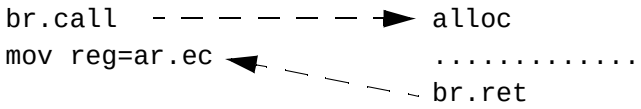
Implication: An application may hang or signal an exception fault under these circumstances. The affected code sequence is not known by Intel to be generated in any current compiled code or exist in any current OS.

Workaround: Separate the predicate producing instruction from its consumer with a stop (as recommended in the *Intel® Itanium® Architecture Software Developer's Manual*, Volume 1: Application Architecture, page 1:157) or change the predication sequence to assure mutually exclusive predication of the instructions in the WAW dependency.

Status: For the steppings affected, see the Summary Table of Changes.

37. Specific code sequence involving br.ret followed by mov from AR.EC may return incorrect EC value

Problem: In the following sequence involving the `mov` instruction (`mov reg=ar.ec`) at the target of a `br.ret` instruction, the general purpose register, which is the target of the `mov` instruction, may receive a stale Epilog Count (EC) value if the `br.ret` is predicted correctly. The stale EC value that is put into the general purpose register is the EC value before the `br.ret` instruction is executed:



Implication: As a result of this erratum, the above code sequence may return an incorrect EC value in the general purpose register. The affected code sequence represents an unexpected usage model for the EC application register used in software-pipelined loops, and is not known to be generated by OSes or compilers.

Workaround: Ensure that a `mov reg=ar.ec` is not the target of a `br.ret` instruction.

Status: For the steppings affected, see the Summary Table of Changes.

38. Specific code sequence involving bsw followed immediately by loadrs may not execute correctly

Problem: Under certain conditions involving a code sequence where the Bank Switch (`bsw`) instruction is followed immediately by the Load Register Stack (`loadrs`) instruction, it is possible that the registers dependent on the `loadrs` instruction execution may not get updated correctly.

Implication: Code that uses the affected sequence may not execute correctly. The `bsw` instruction is a privileged instruction and there is no known usage model involving a `bsw` instruction immediately followed by a `loadrs` instruction.

Workaround: Separate the `bsw` instruction and the `loadrs` instruction with a nop bundle in the program sequence.

Status: For the steppings affected, see the Summary Table of Changes.

39. PAL_TEST_PROC may cause unexpected system behavior

Problem: The PAL_TEST_PROC ‘late FP load/store test’ may overwrite the fr2-fr5 and fr30-fr31 FP registers and the Bank 0 gr16–gr23 general registers may be overwritten by the ALAT, VHPT, TLB and memory attributes tests.

Implication: PAL_TEST_PROC may corrupt the following registers: Bank 0 gr16–gr23 (general registers) and the fr2– fr5, fr30–fr31 (FP registers).

Workaround: Use different registers or save/restore the contents before/after running PAL_TEST_PROC.

Using the self-test control word of the PAL_TEST_PROC procedure, set the following bits to ‘1’: To avoid corrupting the Bank 0 general registers, do not run the ALAT (bit 46), VHPT (bit 35), TLB (bit 33) and mem_attr (bit 44) tests. To avoid corrupting the FP registers do not run the late_fp_ld_st (bit 40) test.

Status: For the steppings affected, see the Summary Table of Changes.

40. FPSWA version 1.12 may overwrite register fr12

Problem: OSs are required to preserve registers fr6 - fr11 on Floating Point SoftWare Assistance (FPSWA) faults. FPSWA version 1.12 may overwrite register fr12 when handling FPSWA faults caused by the `fma`, `fms` and `fnma` instructions consuming denormalized or unnormalized values.

Implication: An OS that saves and restores fr12 in addition to fr6-fr11is not susceptible to this issue. The only known current exposure is with the Linux* OS.

Depending on how an application uses fr12 and how the OS preserves it, this situation could lead to a number of different failure scenarios including incorrect data. This erratum is limited to FPSWA version 1.12.

Workaround: Upgrade to FPSWA version 1.18 or later which corrects the issue.

Status: For the steppings affected, see the Summary Table of Changes.

Specification Changes

There are no Specification Changes for this revision of the *Intel® Itanium® Processor Specification Update*.

Specification Clarifications

There are no Specification Clarifications for this revision of the *Intel® Itanium® Processor Specification Update*.

Documentation Changes

There are no Documentation Changes for this revision of the *Intel® Itanium® Processor Specification Update*.